This project will be divided into 3 different parts

- Front-end  
- Back-end  
- API Layer

Front-end Tech stack: This will be the user interface from where user will do registration and login to their account. This section will be responsible for executing different API’s.  
- ReactJS  
- HTML  
- CSS

Back-end Tech stack: This will be the administration section from where admin can manage users / tenants section, enabling / disabling any external API services and pricing for giving these services.  
 - NodeJS  
 - HTML  
 - CSS

API Tech stack: This will be a part of Back-end system. Basically it will be a layer inside our back-end system. This will be responsible for managing all external API’s. All API’s will pass through this layer.  
 - ExpressJS

**Workflow of entire project:**

* User will interact with the system from front-end. They will login to their account (Register first if not registered), initiate API call which they want. We will be using browser cache ( Local Storage / Cookies ) for data which we want to serve immediately or for which we do not want to call API frequently.
* Whenever user will call any service, after logging in to their account, first that request will go on to the caching server and it will search for the record there. If it finds record there, then the response will be immediately severed back. If not, then the request will be passed to our API layer.
* Our API layer will be composed of many different sub-layers which will be responsible for handling different tasks. Here we will have following layers:  
  **Validation Layer:**  
   Available Credit check  
   Incoming parameter check  
     
  **Factory Layer for service:** This will be a Factory Design Pattern class, which will be responsible for creating object of specific API service (it maybe of AWS, IBM watson, Google or Microsoft)  
     
  **Deducting credit and Notify layer:** Consume (specific) credits from user/tenant account, which we are charging for calling that service. Update their account and notify them by SMS or email.
* Send response back to user first (since we don’t want our user to wait for response) and along with this execute an asynchronous call to our caching server to update our cache.

**Tech Stack:**

ReactJS – It’s a small and light weight JavaScript library, easy to learn and work with for any fresher or experienced developer.  
  
 ExpressJS – Middle ware for adding our routing logic and API management.  
  
 NodeJS – Useful in writing back-end code as it holds asynchronous HTTP call concept. Moreover it is also written in JavaScript, so there will be no need to learn any new language. Same developer, who will be working on front end can also work on back-end.  
  
 Mysql / MongoDB: We can go either with Mysql or MongoDB.  
  
 Redis Cache: Easy to work with. Data storage capacity is very large like 232 key-value pairs can be stored and further more it can be managed with machines hardware. Also there is a TTL (Time To Live) concept which can be used in managing time to hold cache before it expires.  
  
 Browser cache (Local storage / Cookie): This is another type of caching mechanism which we can use whenever we want to store information without sending requests to server.

**Deployment Strategy**:

CI / CD: With Jenkins or Docker (these tools can easily integrate with any application and are useful for CI/CD purpose).

Tool to manage project, code and for code review:  
 For this purpose we can use bitbucket. Project can easily be cloned if someone want a fresh copy to start with. We can push or pull the code with changes. Also we can create branches so that team members can work separately on different modules.  
  
SonarQube: It can be used for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs and security vulnerabilities.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**Training Plan and project management strategy:**

- From the beginning of any project, we can create and maintain documentation which can be similar to this one in the beginning, but as we move further, it will hold each and every information about classes or modules being added on.  
  
- Best practices will be followed in order to maintain coding consistency throughout the project.

- 30–60 minutes session can be given for 3-5 days to train team members so that they can understand in and out of a project and how to work on it.

**Design Representation:**

Call to Relevant Services

Factory Design Pattern Class

Frontend

AWS

Microsoft Azure

Google

IBM Watson

Caching Layer

API Layer

Backend

Frontend

Frontend